Exercises

Ex 1.

Do not use an array to hold numbers that user enters in this exercise!

Write a program that calculates average of positive numbers that user enters. Program asks user to enter numbers and calculates the average of entered numbers when user enters 0 as the number. The zero is not included in the average. If user enters a negative number the program must print a message telling that only positive numbers are accepted and ignore the negative number.

Ex 2.

Write a program that defines a floating-point array of 12 elements. The program then asks user to enter salary of each month and stores the values in the array. After user has entered all monthly salaries, the program calculates the total yearly income.

Finally the program prints all salaries with two decimal precision and prints the total after the salaries.

Example of final print out (three dots indicates rows not shown in this example):

```
month[ 1] =    998.00
month[ 2] =    998.00
...
month[ 6] =   1489.51
...
month[12] =    998.00
total     =  12475.12
```

You need to get familiar with printf field width specifiers to complete this!

Note that square brackets, equal signs, and decimal points must be aligned as shown above!

Ex 3.

Write a function that takes an array of integers and array size as a parameter and returns a float. The function calculates average value of the array and returns it.

Write a program that uses an array to hold course grades. The array is initialized to zero in the beginning. The program asks user how many students are to be graded. Program must not allow values that are greater that the array size.

Then the program asks user to enter a grade for each student. Only values in the range 0-5 are accepted. Student number that is displayed to the user is array index + 45000.

(For example if array index is 2 user is asked to enter grade for student nr: 45002)

Program must check that array bounds are not violated!

In the end program calculates the average of course grades using the average function and prints the average with one decimal accuracy.

Remember to write both function declaration and definition!

Ex 4.

Write a program that computes dot product of two 3 element vectors.

Program asks user to enter initial values for vectors.

Program must have a function to compute the dot product. The function takes the vectors as parameters and returns a float (dot product). Function may not modify source vectors.

Program must print the initial values before multiplication and result after multiplication.

The dot product function must do only the dot product computation and may not print anything.

Ex 5.

Write a function that converts all white space in a string to underscores.

Write a program that asks user to enter a string. When user has entered a string, the program calls the function above and prints the string. The program keeps asking for strings until user enters "stop" or "STOP". When user enters stop the program terminates.

Note. To avoid platform dependent problems with string use strncmp and compare only the first four characters to detect stop.

Ex 6.

Write a program that asks user to enter current time in 24 hour format and asks how long you want to sleep. Program then calculates your wake up time and prints it. The program must check that values are entered in correct format and must print an error message if invalid input is entered. Program must also check that the entered numbers are in valid range (0 - 23 / 0 - 59).

Program must always produce valid times as output – it must roll minutes over to next hour correctly and roll over midnight correctly.

Example:

```
Enter current time (hh:mm): 21:56

How long do you want to sleep (h:mm): 8:30

If you go to bed now you must wake up at 6:26.
```

Ex 7.

Write a function that converts the first letter of every word in a string to uppercase.

Write a program that asks user to enter a string and then calls the function above. The program keeps asking for strings until user enters "stop" or "STOP". When user enters stop the program terminates.

Ex 8.

Write a function that performs a simple cipher on the given string. The cipher maps letters of the alphabet A-Z to reversed alphabet: A is mapped to Z, B is mapped to Y, C is mapped to X, etc. The function needs to preserve the case of each letter. If letter is upper case, it must be upper case after encryption and same with lower case letters. Only letters A-Z are affected by the cipher, all other characters must remain unchanged.

Due to the nature of this “cipher” if the function is applied twice in a row on the same string the result should be the original string..

Write a program that uses the function above to encrypt and decrypt a given text. The program must ask user to enter a string. Then program prints the original string, encrypts it and prints the resulting string, encrypts the string again and prints it. The string that is printed after the second encryption should be same as the original string.

Hint. You can map the characters by using a constant string that contains reversed alphabet. To encrypt a letter, you index the array with the letter to encrypt.

Ex 9.

Write a function that takes a pointer to student information structure and a string (city) as a parameter and returns an integer. The function prints all students that live in the given city. The information must be printed in formatted columns where text is aligned to left and numbers are aligned to right. The array of student structures passed as parameter is assumed have a terminating entry where the student number is zero. The terminating element must not be printed. The function returns the number of students that were printed.

The student information structure must contain the following information:

- Name of the student
- Student number
- Street address
- City
- Total number of credits

Write a program that defines a 15-element array of students and initializes the array with an initializer that contains at least 7 students and the terminating entry.

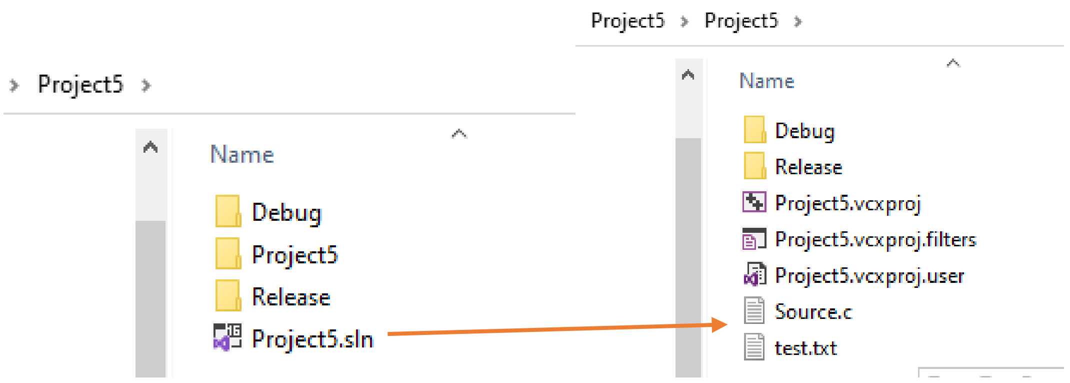
The program asks user to enter a city and prints the students using the function described above. If no students were printed the program must print: “No students live in the given city”. Then program asks user to enter another city. Program can be stopped by entering “stop” as the name of the city.

Notes about file locations with different IDEs

When you open a file without a path the file is opened in the current directory, from the perspective of the running program. Which directory your IDE uses varies. Typically, it is located somewhere in the project directory or its subdirectory.

Visual studio

In visual studio the directory where the file should be is the one where your source files are.



When you open a file **without a path** the file is opened in the project directory.

How to find where the files should be?

Write a program that opens a file with an easily recognizable name, eg. “my_best_test.txt”, for writing and then closes it. Run the program. Browse your project folders to see where the file was created. The folder that the file was created in is the one where the files without a path should be.

Ex. 10

Write a program that defines an array of 20 integers. Then program asks user to enter numbers and to enter a negative number to stop. Program keeps asking more numbers until a negative number is entered or 20 numbers have been read. Then program opens text file “numbers.txt” for writing, writes the entered numbers to the file, and closes the file. All numbers must be separated by white space.

Ex. 11

Write a program that reads numbers from a text file into an array of 100 integers. The program opens a text file and reads numbers from the file into the array until file ends of 100 numbers have been read. Then program closes the file.

After reading numbers program calculates average of the numbers and opens another file for writing. On the first line program writes the count of numbers and the average. Starting from the second line program writes the numbers, one per line, and closes the file.

Ex. 12

Write a program that defines a structure for price information. The structure must contain name and price of the item. Program must define an array of structures and an integer to hold the count of items. The size of the array must be at least 20 elements and program must keep track of the actual count and ensure that array bound are not violated.

The program asks user to enter item information: name first, “stop” is used to stop entry.

When user stops entering price information the program opens “pricelist.txt” for writing and writes the content of the array to the file.

Each item is stored on a single line. The price comes first and is separated from the name with a semicolon. For example:

```
9.99;USB mouse

19.98;SSD 128GB

589.90;MSI GeForce RTX 2070 SUPER VENTUS OC

59.90;HyperX Alloy Core RGB
```

Ex. 13

Write program that reads shop inventory from a file to an array that can hold information about 100 items. Information about each item is stored in a structure that contains the following information:

- Name of the item
- Price of the item
- Number of items in store

Each item is stored on a single line. The price comes first, then the name of the item and number of items in store. Each member is separated by a semicolon. For example:

- `9.99;USB mouse;25`
- `19.98;SSD 128GB;13`
- `589.90;MSI GeForce RTX 2070 SUPER VENTUS OC;3`
- `59.90;HyperX Alloy Core RGB;194`

The program must ask user to enter the filename and then read items until the end of file or up to 100 items which ever happens first. After reading the file the program must print the number of lines read and total number of items in store (=sum of number of items).

Create couple of test files manually and test that you program works.

Ex. 14

Write a program that stores student information to a file in binary mode.

The student information structure must contain the following information:

- Name of the student
- Student number
- Street address
- City
- Total number of credits

The program asks if user wants to add students or print existing student.

If user chooses to print existing student the program opens file “students.dat” for reading in binary mode and if file was opened successfully reads and prints student structures until the end of file occurs and then closes the file.

If user chooses to add students the program opens “students.dat” for append in binary mode and then asks user to enter student information, and writes the structure to the file. When user enters “stop” as student name the program closes the file.

Ex. 15

Write a program that reads a file in binary mode and calculates a 16-bit CRC of the data. The program must ask user to enter the filename.

When all data has been read the program prints the name of file, the number of bytes read and crc. CRC must be printed as a 4-digit hexadecimal number (see printf formatting).

When the file information has been printed the program must close the file and ask user to enter another file name or “stop” to exit the program.

Use following code for CRC-calculation (adapted from: https://stackoverflow.com/questions/10564491/function-to-calculate-a-crc16-checksum)

```
uint16_t crc16(const uint8_t* data_p, unsigned int length) {
    uint8_t x;
    uint16_t crc = 0xFFFF;

    while (length--){
        x = crc >> 8 ^ *data_p++;
        x ^= x>>4;
        crc = (crc << 8) ^ ((uint16_t)(x << 12)) ^ ((uint16_t)(x <<5)) ^ ((uint16_t)x);
    }
    return crc;
}
```

Ex. 16

Write a program that takes filename and a number as parameters. The program opens the file in text mode and prints the file content starting from the line given as parameter. If the file does not have that many line the program prints a message: “file <filename from command line> not have <number> lines”.

Program must check if the file was opened successfully and must check that correct number and type of parameters was given on the command line. If there is an error in parameter format or number of parameters the program must print brief instructions what parameters should be given before exiting.

Ex. 17

Write a program that takes filename and number as parameters. The program opens the file in binary mode and splits the file into smaller files where the size of the smaller file at maximum the number given as parameter. The name of each file created is the name of the original file suffixed with “.part” and a three-digit number starting from one with leading zeros.

For example, if the user gives the following parameters:

```
example.txt 200
```

If we assume that example.txt is 782 bytes long the program creates following files:

```
example.txt.part001

example.txt.part002

example.txt.part003

example.txt.part004
```

The length of the first three files is 200 bytes and the last file is 182 bytes long.

Ex. 18

Write a program that asks user to enter a hexadecimal number. See scanf documentation for information how to read a hexadecimal number. When user has entered the number program prints it with 8 hexadecimal digits and leading zeros.

Then program asks user to enter an integer in the range from 0 to 31.

Program then places the integer in the first number on bit positions 6-10 of the first number without modifying the other bits and prints the result in hexadecimal with leading zeros.

Hint: You need to clear bits 6-10 from the number before placing the other number in that position.

Note: bit numbering starts from 0

Ex. 19

Write a program that allows user to remove bytes from a binary file.

Program asks user to enter file name and then displays the file size and asks user what to remove. User enters offset from the beginning of the file and number of bytes to remove. Program then overwrites the original file with the data where the user specified bytes are removed. Program must display an error message if user tries to remove data past the end of the file or if the offset is invalid.

Ex. 20

Write a password generator **function** that takes three parameters: character pointer (array), integer (size of the array), a const char pointer (a word). The function generates password and stores it in the array. The password must contain the word given as a parameter reversed at a random location and the rest of the password contains random **printable** characters. Note that printable characters are not limited to alphanumerical characters. Hint: fill the password first with random characters and then copy random word.

Write a program that asks user to enter a word to place in the password or to enter “stop” to stop the program. If user does not enter stop the program asks user to enter password length in the range of *length of word + 2* to 30. Then program generates a password using the generator functionand then prints it. Then the program asks for another word.

For example:

User enters: metropolia, 15

Program prints: #8QailoportemGe

Ex. 21

Write a program that asks user to enter name of a file. The program opens the file in text mode and calculates frequency of first letters of all words. Only words beginning with a letter are counted. Finally, the program prints a list of five most frequent first characters and their frequencies.